

Bhartiya Vidya Bhavan’s

Sardar Patel Institute of Technology, Mumbai-400058

Department of Electronics and Telecommunication Engineering

# IT424: Blockchain Technology and Applications

**Lab-7: Blockchain and Cybersecurity Part-I**

**Develop a blockchain application for Cybersecurity.**

# Name: Shubham Golwal UID: 2020300015

**Objective:** **Develop a blockchain application for Cybersecurity**

**Outcomes:** After successful completion of lab students should be able to

Implement an Ethereum private blockchain

Build two-factor authentication (2FA) using Blockchain

Write a smart contract using Solidity Language

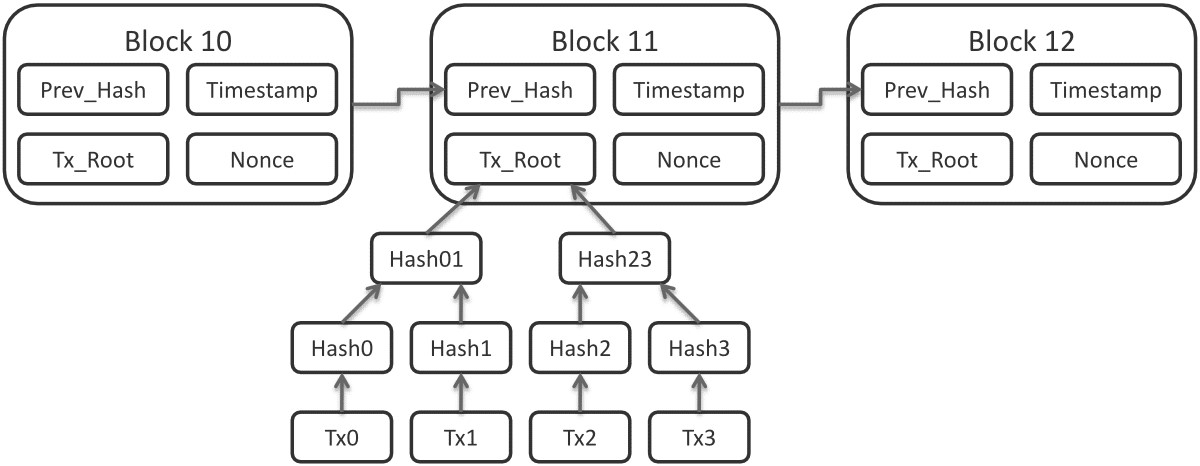
Compile and run the 2FA using Ethereum Blockchain

Use REST API and Flask microframework

**System Requirements:**

PC (C2D, 8GB RAM, 100GB HDD space and NIC),Ubuntu Linux 14.04/20.04

Internet connectivity,Python Cryptography and Pycrypto,Nodejs, Truffle,Ganache-cli , solidity,REST API

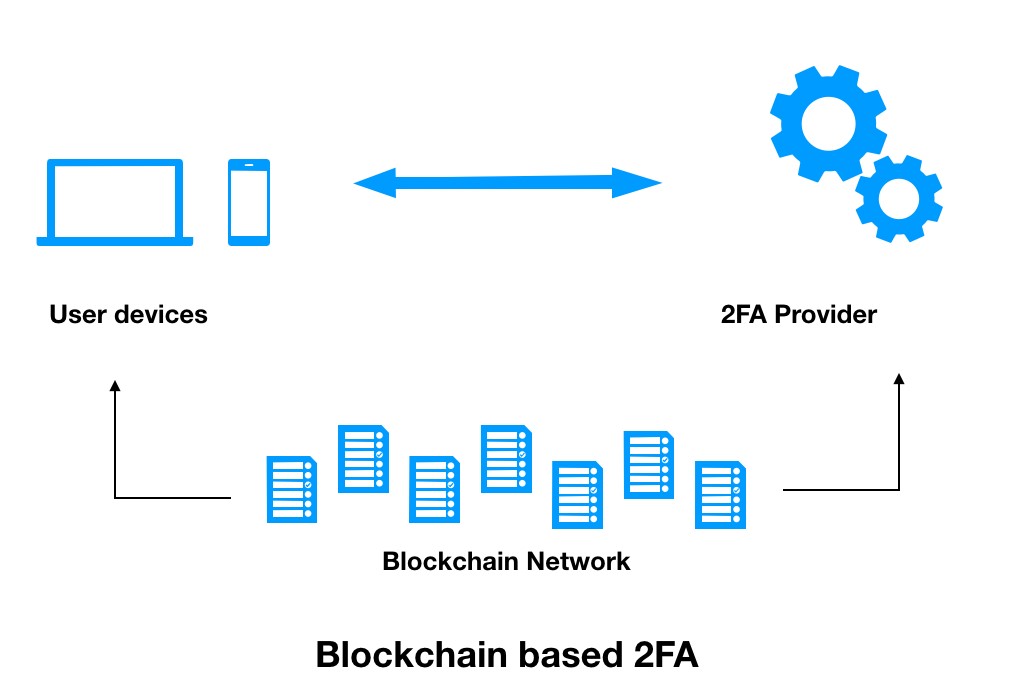


**Figure-1: Blockchain Implementation**

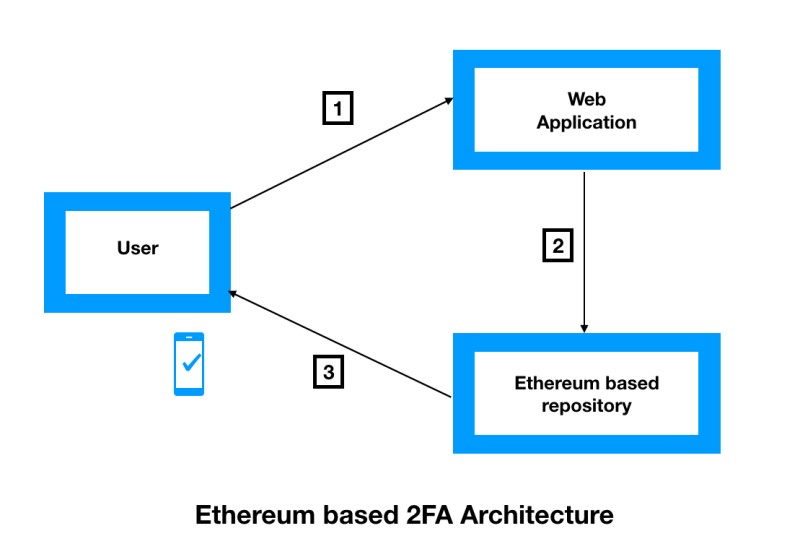
1

# Part-I: Two-Factor Authentication with Blockchain

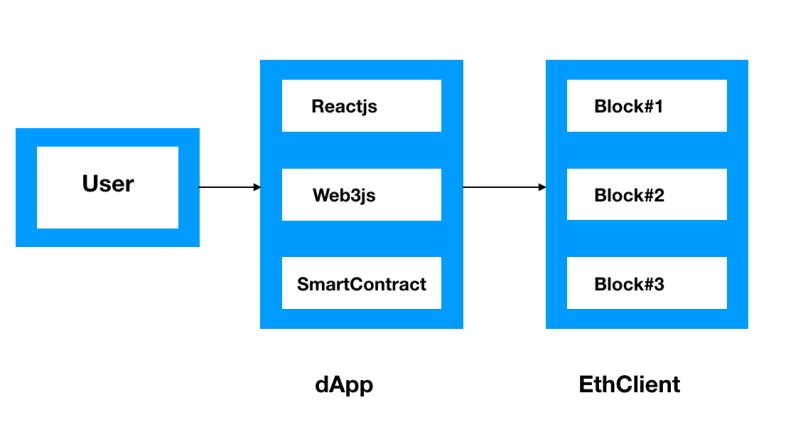
Two-factor authentication (2FA) provides an added layer to the existing credential-based system protection as a solution to this drastically growing problem.



# Figure-2: Blockchain-Based 2FA



# Figure-3:Solution Architecture



# Figure-4: Ethereum based 2FA

**Procedure:**

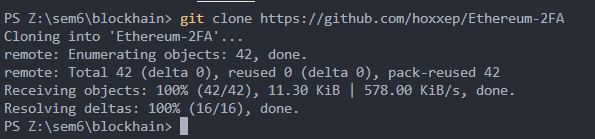
1. Create a directory under BTA

On windows we create directory named Blockchain inside this we will clone the repo and write all our code



1. Clone or download the Ethereum-2FA

$ git clone https://github.com/hoxxep/Ethereum-2FA



$cd Ethereum-2FA



The files in the preceding screenshot are explained as follows: **contracts:** This folder includes our smart contract, TwoFactorAuth.sol



**migrations:** This folder consists of migration files to deploy the contract to the blockchain



**test:** This folder consists of server.js, which is responsible for event authentication in our contract



**node\_modules:** This folder includes all the libraries

**truffle.js:** This configuration file consists of a set of configurations to connect to the blockchain



**package.json:** This is where we specify a configuration of our projects, such as name and scripts

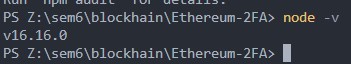


1. Install Nodejs

I have nodejs pre installed on my laptop

#Check Nodejs version

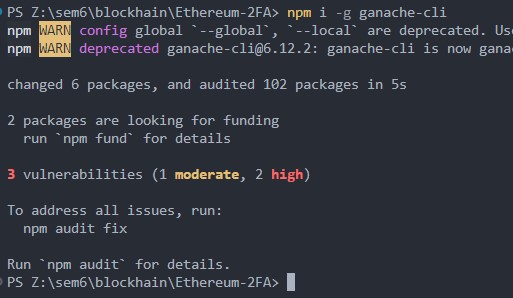
$node -v



#Turning up Ethereum

#Install ganache-cli

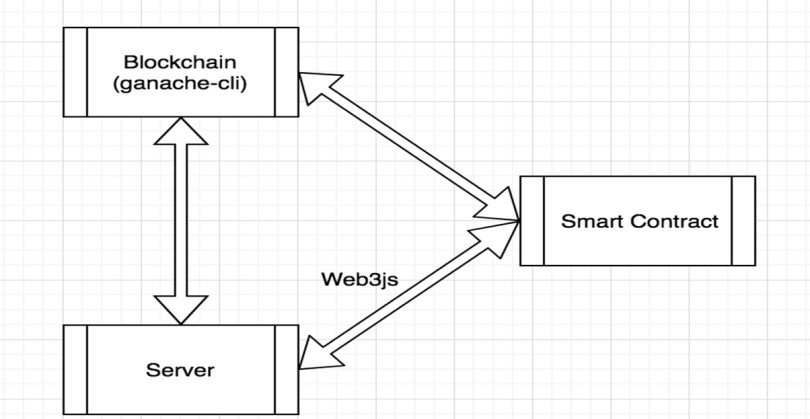
$npm install -g ganache-cli



# Components

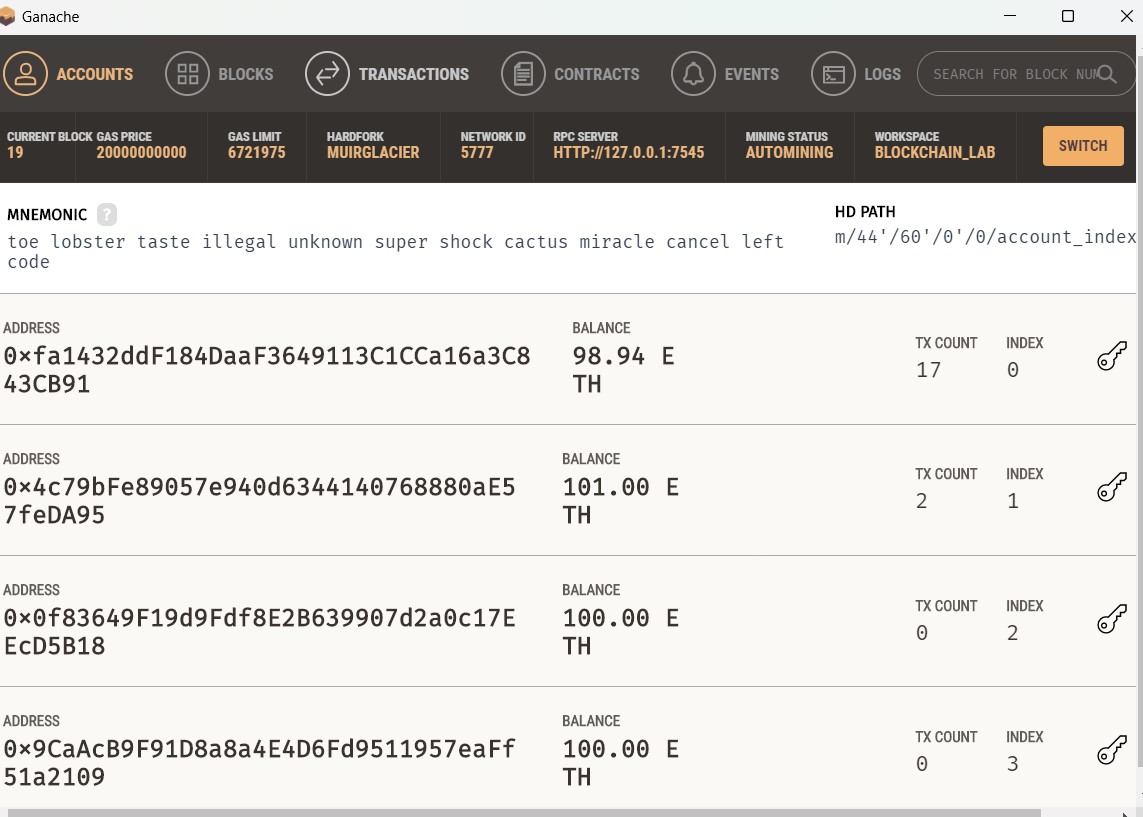
The following are the three core components of this project, shown in the following diagram:

* A blockchain network (which we will develop with the Ganache CLI)
* A smart contract
* A server communicating with the blockchain

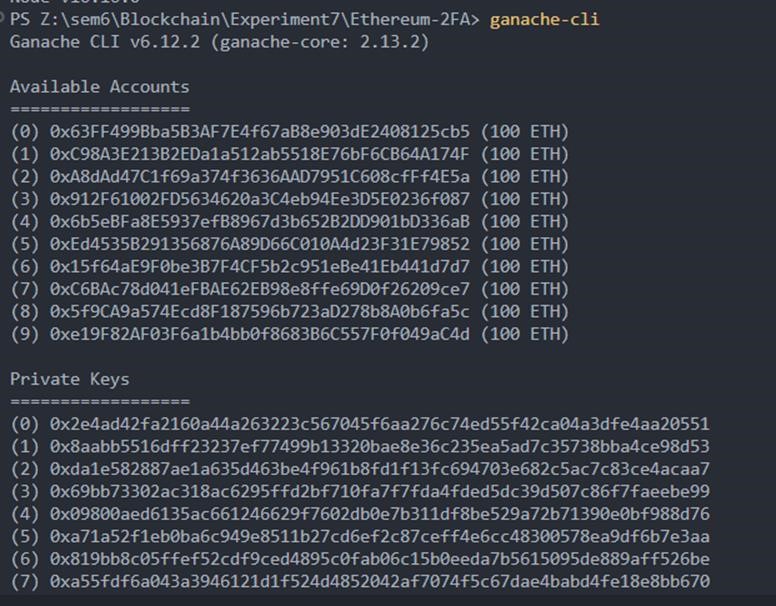


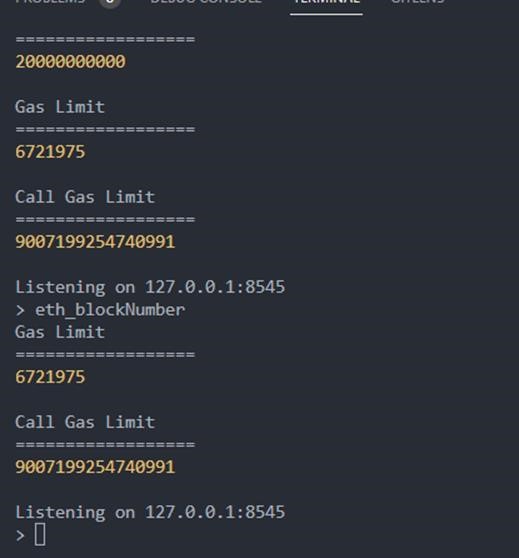
# Figure-5: Components of Blockchain-Based 2FA

$ganache-cli



Start the ganache-cli to listen on port 8545

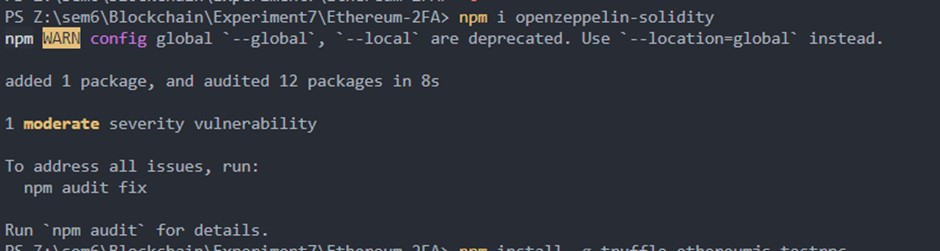


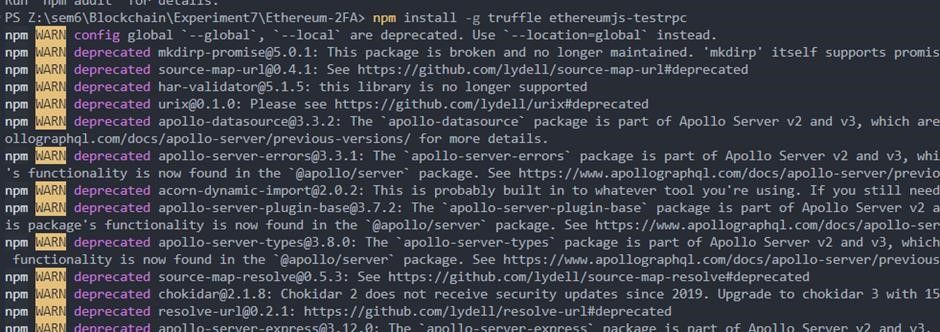


# Turning up the smart contract

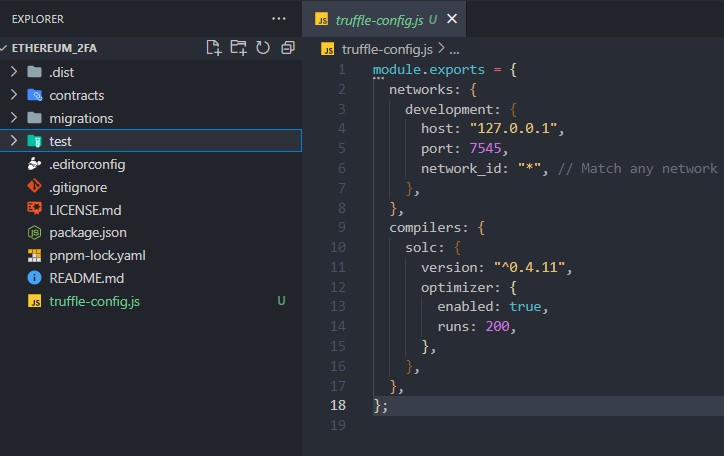
$cd Ethereum-2FA

Install the necessary dependencies

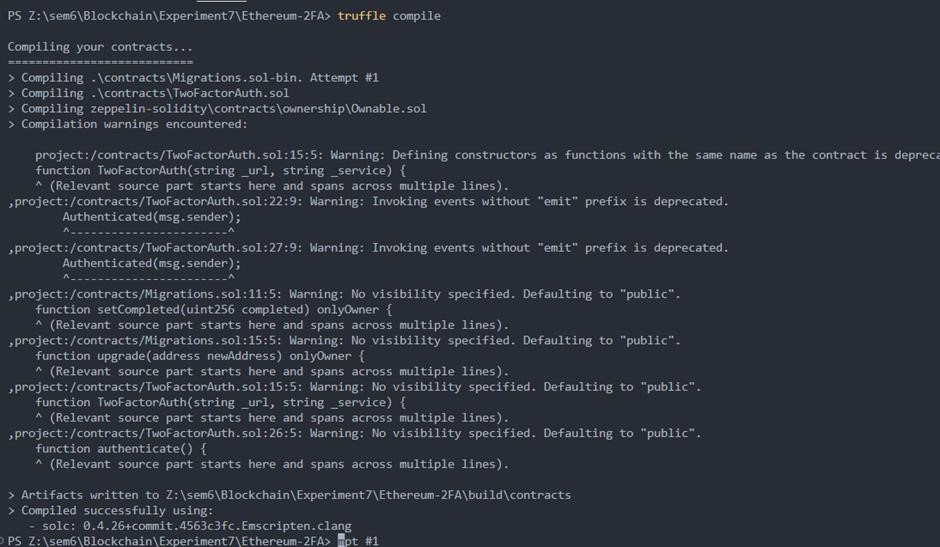




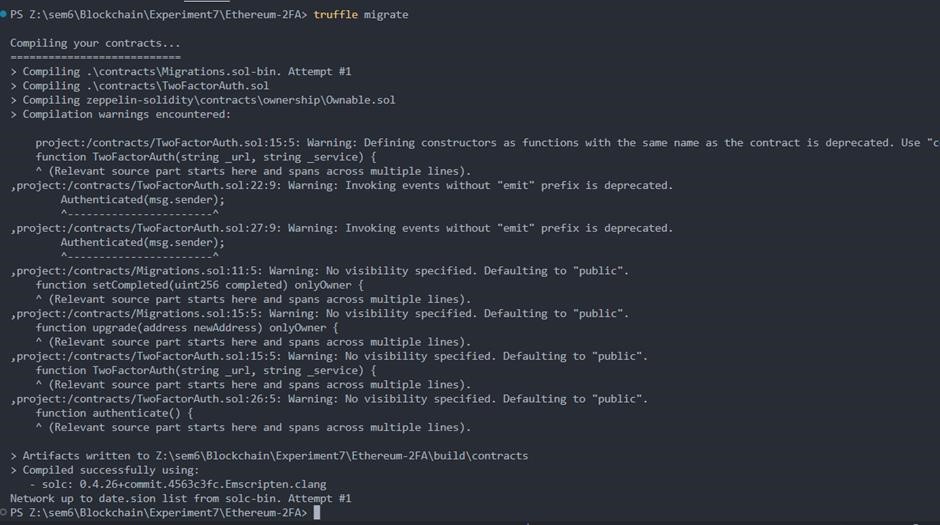
Update truffle.js to truffle-config.js



Run the truffle compile command

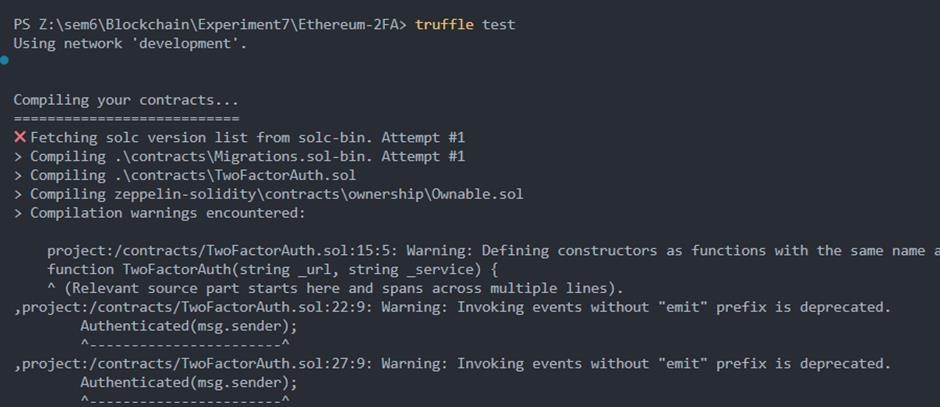


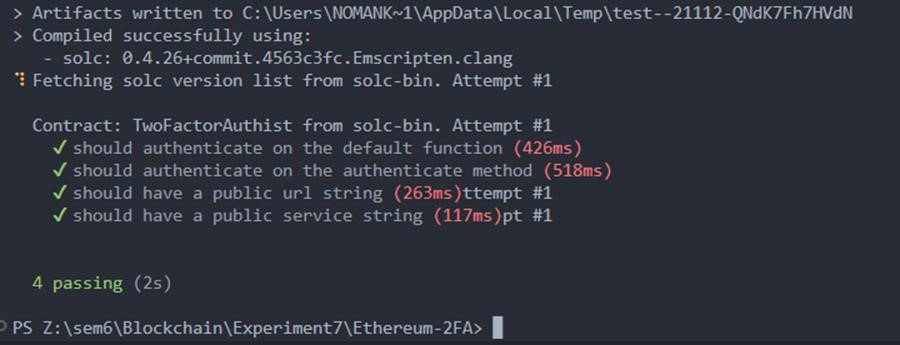
Run truffle migrate command



$truffle test

Now let's test our contracts using the truffle test command

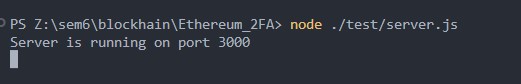




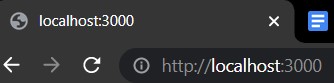
# Figure-8:Ethereum-2FA

Server Code in express js with web3

|  |
| --- |
| require("./helpers/assert"); const express = require("express"); const app = express(); const port = 3000;  const bodyParser = require("body-parser"); const Web3 = require("web3");  const web3 = new Web3("http://127.0.0.1:7545"); const contract = require("truffle-contract"); const twoFactorAuth = contract(  require("../build/contracts/TwoFactorAuth.json")  );  twoFactorAuth.setProvider(web3.currentProvider); const twoFactorAuthInstance = twoFactorAuth.deployed();    app.use(bodyParser.json());  app.use(bodyParser.urlencoded({ extended: true }));    app.get("/", async (*req*, *res*) => { try {  const instance = await twoFactorAuth.deployed(); const accounts = await web3.eth.getAccounts();  const result = await instance.authenticate({ from: accounts[1] }); const events = result.logs; |
| res.status(200).send(`${events[0].event} : ${events[0].args.\_user}`);  } catch (error) {  return res.status(500).send({ message: "Internal server error" });  }  });  app.listen(port, () => {  console.log(`Server is running on port ${port}`);  }); |



# Open Google Chrome and access the localhost on port 3000



Successful authentication:



**Conclusion:** In this experiment,

1. Created a private Ethereum blockchain by configuring network parameters and utilizing Ethereum clients.
2. Developed a 2FA system using Solidity language and a smart contract, which was deployed and executed through the Ethereum network.
3. The Ethereum Blockchain can compile and run this security solution, providing robust and reliable security.
4. Technical knowledge and expertise are required to implement these procedures effectively.

**References:**

1. Two-factor authentication through an Ethereum contract.

<https://github.com/hoxxep/Ethereum-2FA>

1. How To Install Node.js on Ubuntu 20.04

[How To Install Node.js on Ubuntu 20.04 | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04)